Continuation Team

JVault

Security Assessment

April 6, 2024

JVault

The security assessment was prepared by Continuation Team.

Subject of Security Assessment:

- **[pool_admin]** - Contract for deploying staking pools and distributing protocol fees received from staking pools.
- **[floating_staking_collection]** (nft-collection) - Contract implementing the mechanics of a floating staking pool. Implemented as an NFT collection (TEP-62) to create a distributed system of contracts.
- **[fixed_staking_collection]** (nft-collection) - Contract implementing the mechanics of a fixed staking pool. Implemented as an NFT collection (TEP-62) to create a distributed system of contracts.
- **[nft_item]** (nft-item) - Contract for storing information about a position in a staking pool, as well as for managing the position. Implemented as an NFT-item (TEP-62).
- **[sharecoms]** - Contract for distributing protocol fees among holders of positions in a specific staking pool.

Codebase:

- **base** - 4c48aab04d559b9aacf9c945a896524a56d83a6b, 8b801fb0d8715f8d47ab5e862109cc1042ddae1d
- **update** - c4843b8622fb1c9a0cfc44581e960f60d881a06f, 63cbc8137a7dca0269bcdd9089a9dcc37acbb727

AUDIT SCOPE

- floating-staking-contracts/contracts/floating_staking_collection.fc
- floating-staking-contracts/contracts/nft_item.fc
- floating-staking-contracts/contracts/pools_admin.fc
- floating-staking-contracts/contracts/sharecoms.fc
- floating-staking-contracts/contracts/imports/constants.fc
- floating-staking-contracts/contracts/imports/utils.fc
- fixed-staking-contracts/contracts/fixed_staking_collection.fc
- fixed-staking-contracts/contracts/nft_item.fc
- fixed-staking-contracts/contracts/pools_admin.fc
- fixed-staking-contracts/contracts/sharecoms.fc
- fixed-staking-contracts/contracts/imports/constants.fc
- fixed-staking-contracts/contracts/imports/utils.fc

FINDINGS

ID	Title	Status
1	Lack of `end_parse` invocation in multiple functions, message schema not validated.	Acknowledged
2	Message parsing when calling `transfer_ownership` does not conform to TEP-62 schema.	Resolved
3	Lack of forced bounce upon unsuccessful action phase handling when calling `transfer_ownership`.	Resolved
4	Message parsing when receiving `transfer_notification` does not conform to TEP-74 schema.	Resolved
5	Checks when receiving `transfer_notification` do not guarantee return of jettons.	Resolved
6	Potential abuse opportunity in jetton-wallets setup.	Resolved
7	Message with `op::report_storage_data` does not ensure user fund return in case of failure on the receiving contract.	Resolved
8	Input data not validated for contract initialization when receiving `op::deploy_new_pool`.	Resolved
9	Response during `nft_item` contract initialization does not match TEP-62 schema.	Resolved

ID	Title	Status
10	Gas payment sufficiency checks do not use gas price calculation using blockchain configuration.	Acknowledged
11	Unsafe usage of hashmap in `sharecoms`.	Acknowledged

RECOMMENDATIONS

ID	Title	Status
-	Recommendation from the **Continuation Team**	Resolved

1 Lack of `end_parse` invocation in multiple functions, message schema not validated.

Locations:

- **floating-staking-contracts/contracts/floating_staking_collection.fc**
- **floating-staking-contracts/contracts/nft_item.fc**
- **floating-staking-contracts/contracts/pools_admin.fc**
- **floating-staking-contracts/contracts/sharecoms.fc**
- **fixed-staking-contracts/contracts/fixed_staking_collection.fc**
- **fixed-staking-contracts/contracts/nft_item.fc**
- **fixed-staking-contracts/contracts/pools_admin.fc**
- **fixed-staking-contracts/contracts/sharecoms.fc**

Status: **Acknowledged**

Description

The absence of `end_parse` leaves us uncertain that the incoming message conforms to the working schema of this contract.

Recommendation

It is recommended to add `end_parse` every time parsing of the incoming payload ends, unless, of course, it should end with a slice according to the scheme.

2 Message parsing when calling `op::transfer_ownership` does not conform to TEP-62 schema.

Locations:

- **floating-staking-contracts/contracts/nft_item.fc#82**
- **floating-staking-contracts/contracts/nft_item.fc#84**

Status: **Resolved**

Description

When calling operations `op::transfer_ownership`, the skipping of `custom_payload` is incorrectly implemented, as well is incorrectly implemented the validation of the ”tail” message schema containing `forward_payload`.

Recommendation

To skip `custom_payload`, it is recommended to use the `skip_dict` function. When validating the schema of `forward_payload`, it is necessary to read `Either`, then ensure that there are no extra cells/bits using the `slice_bits_refs` function.

3 Lack of forced bounce upon unsuccessful action phase handling when calling `op::transfer_ownership`.

Locations:

- **floating-staking-contracts/contracts/nft_item.fc#98**
- **floating-staking-contracts/contracts/nft_item.fc#102**

Status: **Resolved**

Description

In some cases, when calling operation `op::transfer_ownership`, an error may occur in the action phase, resulting in the loss of user funds directed towards paying fees in the transaction chain.

Recommendation

To prevent the loss of user funds, it is recommended to include the +16 flag when sending messages. This flag enables bounce in case of errors in the action phase.

4 Message parsing when receiving `transfer_notification` does not conform to TEP-74 schema.

Locations:

- **floating-staking-contracts/contracts/floating_staking_collection.fc#149**

Status: **Resolved**

Description

In the received message with `op::transfer_notification`, the `Either` bit in `forward_payload` is ignored, violating the TEP-74 standard.

Recommendation

Before parsing the `forward_payload` body, it is necessary to check the `Either` bit. If necessary, load the next cell containing the body.

5 Checks when receiving `transfer_notification` do not guarantee return of jettons.

Locations:

- **floating-staking-contracts/contracts/floating_staking_collection.fc**
- **floating-staking-contracts/contracts/pools_admin.fc**

Status: **Resolved**

Description

The use of `throw`, `throw_if`, `throw_unless` functions when receiving jettons may result in the loss of user funds.

Recommendation

It is recommended to perform checks as conditional statements and implement a refund function if the conditions are not met.

Additionally, for exception handling, the `try-catch` construct or a custom exception handler written to register `c2` can be used.

6 Potential abuse opportunity in jetton-wallets setup.

Location: **floating-staking-contracts/contracts/floating_staking_collection.fc#L155-L162**

Status: **Resolved**

Description

The ability to set `lock_wallet_address` and `rewards_wallet_address` during the first jetton transfer to `floating_staking_collection` allows dishonest users to disrupt the service by setting invalid jetton wallets before a transaction arrives with `op::take_wallet_address` (TEP-89), containing the address of a valid jetton wallet.

Recommendation

It is recommended to remove this part of the code. If it is necessary to support jettons issued before the TEP-89 standard, then consider adding an operation to set `lock_wallet_address` and `rewards_wallet_address`, accessible only to the pool administrator.

7 Message with `op::report_storage_data` does not ensure user fund return in case of failure on the receiving contract.

Location: **floating-staking-contracts/contracts/nft_item.fc**
Status: **Resolved**

Description

When calling the operation `op::get_storage_data` in `nft_item`, messages with `op::report_storage_data` are sent. If the execution of the receiving contract fails, the funds will return to the `nft_item` contract instead of the user.

Recommendation

It is recommended to add handling of bounced transactions in the `nft_item` contract, redirecting the funds to the user's wallet.

8 Input data not validated for contract initialization when receiving op::deploy_new_pool.

Location: **floating-staking-contracts/contracts/pools_admin.fc#L177-L193**

Status: **Resolved**

Description

When calling the operation `op::deploy_new_pool` in `pools_admin`, input data for initializing the staking pool contract is not validated. This could result in the creation of a non-functional staking pool.

Recommendation

It is recommended to validate the input data to ensure it meets the requirements for initializing the staking pool contract.

9 Response during `nft_item` contract initialization does not match TEP-62 schema.

Location: **floating-staking-contracts/contracts/nft_item.fc#L142-L149**

Status: **Resolved**

Description

During the initialization of the `nft_item` contract, a message with `op::ownership_assigned_nft` is sent to the initializer's address. This operation complies with the TEP-62 standard; however, the message schema does not match the one presented in the standard.

Description

It is recommended to supplement the message with the necessary information to align it with the TEP-62 standard. Alternatively, a separate operation code for messaging the initialization result of `nft_item` can be implemented.

10 Gas payment sufficiency checks do not use gas price calculation using blockchain configuration.

Locations:

- **floating-staking-contracts/contracts/imports/constants.fc#L25-L39**
- **fixed-staking-contracts/contracts/imports/constants.fc#L26-L40**

Status: **Acknowledged**

Description

Constant values in TON are used for verifying the sufficiency of funds for gas payment and other expenses. Changes in the blockchain configuration can lead to an increase or decrease in gas prices, potentially resulting in contract malfunction or the need to send more TON than necessary.

Recommendation

It is recommended to store gas values in constants and use the `get_compute_fee` function during contract execution for gas fee calculation. Additionally, utilize the `get_forward_fee` function for forward fee calculation when message size remains constant.

11 Unsafe usage of hashmap in `sharecoms`.

Locations:

- **floating-staking-contracts/contracts/sharecoms.fc**
- **fixed-staking-contracts/contracts/sharecoms.fc**

Status: **Acknowledged**

Description

The `sharecoms` contract utilizes 2 hashmaps with large keys. However, there is no provision for clearing the hashmap data. In the event of adding a large number of entries to the hashmap (e.g., in a staking pool with a large number of users), the contract's state size may exceed the limit, causing the contract to cease functioning.

Recommendation

It is recommended to reconsider the contract architecture, avoid using hashmaps, and implement a distributed system of contracts.

Alleviation

[Project Team]: We have added a withdrawal function to the `sharecoms` contract, which becomes available when the hashmap reaches a certain size. The contract will not cease functioning entirely, and the funds will not become locked.

This solution is suitable for temporary use.

Recommendation from the Continuation Team

Consider adding code to the `floating_staking_collection` and the `floating_staking_collection` contracts that enables the withdrawal of ”stray” TON and jettons. This enhancement will allow for rectifying situations where tokens become stuck in these contracts. This is crucial because it prevents the loss of valuable tokens due to unforeseen circumstances.

Good example.

Status: **Resolved**

Conclusion

A security assessment of the **JVault** project revealed several issues across five smart contracts: **pool_admin**, **floating_staking_collection**, **fixed_staking_collection**, **nft_item**, and **sharecoms**.

The identified problems included lack of message schema validation, necessity for forced transaction rollbacks, non-compliance with TEP standards, and potential abuse opportunities. However, all identified issues were acknowledged and accordingly addressed.

The recommendations provided by the audit team, including gas payment sufficiency checks and safer usage of hashmaps, have been acknowledged and are being implemented to enhance the security and functionality of the contracts.

The security of the **pool_admin**, **floating_staking_collection**, **fixed_staking_collection**, **nft_item**, and **sharecoms** contracts is currently at a fairly good level.

Continuation
Team

2024